TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 10508 • LUBBOCK, TEXAS 79408
U.S. Consumer Products Group

April 24, 1983

MCC Powers
2942 MacArthur Blvd.
Northbrook, IL 60062
Attn: John Jeffers

Dear Sir,

Enclosed you will find listings of code that illustrate the use of a CC40 as a peripheral device. The program is called OTRAN and is used to transfer a BASIC program between two CC40s that are connected directly together with a HEX-BUS cable. This package includes:

User guide and operation instructions
Complete program listings
Cartridge link map with program addresses
Complete set of equates for use with program

If you have a question regarding this example, please contact me at 214-997-2454.

Sincerly,

Bud

Bud Gerwig
Texas Instruments
8390 LBJ Freeway
Dallas, TX 75243
214-997-2454
M/S 3662

/lb
enclosure
cc: Ross Deem
[redacted]
Randy Marker

Fifty Years of Innovation

Compact Computer Products
CC-40 BASIC Program Transfer Utility
OTRAN User's Guide

Compact Computer Group
April 5, 1984
Revision 1.0

SECTION 1

OTRAN User's Guide

1.1 Scope of Usage

This software package can transfer both normal and compressed image BASIC programs from one CC-40 to another across the HEX-BUS. It cannot transfer assembly language or other language programs.

1.2 Terms

Throughout this guide, the following terms will be used to describe various parts of the BASIC Program Transfer Utility, OTRAN.

OTRAN -This software package, OTRAN meaning the TRANsfer of programs between CC-40s using the OLD statement.

Master -The CC-40 containing the program to be transferred and the CC-40 containing the OTRAN cartridge

Slave -The CC-40 which. is to have the BASIC program transferred to it from the Master.

1.3 Using OTRAN

The following steps should be used to operate the OTRAN software system.

Step #1 -Turn off both the Master and Slave.

Step #2 -Install the OTRAN cartridge in the Master.

Step #3 -Turn on the Master.

Step #4 -When the Master responds with a 'Running' message, connect the Master to the Slave.

Step #5 -Turn on the Slave.

Step #6 -Type in 'OLD "60"' on the Slave.

Step #7 -Press ENTER on the Slave.

Step #8 -When the Slave block cursor comes back, turn off the Slave.

Step #9 -Disconnect the Slave from the Master.

Step #10-If you are finished, press BREAK on the Master, else, continue at step #4.

Step #11-When the Master responds with 'Program stopped. Press any key.', press any key.

Step #12-When the Master display clears, turn off the Master.

Step #13-Turn on the Slave, and proceed.

NOTE

Make sure that both Master and Slave are disconnected from one another until it is time to connect them.

1.4 Completion

The program that was resident in the Master's program area is now copied into the program work area of the Slave unit, and is ready to run.

1.5 Other features

OTRAN can be stopped at any time after it says that it is running. The way to stop it is to press the BREAK key, and wait for OTRAN to respond with a program stopped message. At this point, pressing any other key will terminate OTRAN. To re-start OTRAN, merely turn off the CC-40 and turn it back on, or enter 'RUN "OTRAN"'.

During data transfers between the Master and Slave, the I/O indicators on both units should be <u>ON</u>. If they are not, disconnect and turn off both units, and start at step #3.

```
DX/9900 LINKER   VERSION 1.0.0  81.197  4/12/84   7:56:20                PAGE
COMMAND LIST

TASK    OTRAN
FORMAT ASCII
PROGRAM 09000
;
INCLUDE WHITE2.SLR.OBJ.MODHEAD
INCLUDE WHITE2.SLR.OBJ.XMITRCV
INCLUDE WHITE2.SLR.OBJ.MESSAGE
INCLUDE WHITE2.SLR.OBJ.OPERHND
INCLUDE WHITE2.SLR.OBJ.CONTROL
;
END
```

```
LINK MAP

CONTROL FILE = WHITE2.TRAN.OTRAN.OTRANLNK

LINKED OUTPUT FILE = WHITE2.SLR.OBJ.OTRANLNK

LIST FILE = WHITE2.SLR.LST.OTRANLNK

OUTPUT FORMAT = ASCII
```

```
PHASE 0     OTRAN      MODULE  ORIGIN = 0000    LENGTH = 0000

MODULE     NO   ORIGIN    LENGTH      TYPE         DATE       TIME      CREATOR

MODHED      1    9000*     000C     INCLUDE      4/12/84    7:54:39    ASMMLP
XMTRCV      2    900C*     0050     INCLUDE      4/12/84    7:55:24    ASMMLP
MESAGE      3    905C*     0073     INCLUDE      4/12/84    7:54:15    ASMMLP
OPRHND      4    90CF*     0125     INCLUDE      4/12/84    7:54:59    ASMMLP
CONTRL      5    91F4*     0148     INCLUDE      4/12/84    7:53:41    ASMMLP



                               D E F I N I T I O N S

 NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

 ABORT    92B8*  5     CONTRL   92BB*  5    *CTRAN    91F4*  5     OPRHND   90CF*  4
 PHEAD    933B*  5     RCV      903B*  2     RCVMSG   905C*  3     SNDMSG   9080*  3
 XMIT     900C*  2

LENGTH OF REGION FOR TASK                = 0000

NUMBER OF RECORDS FOR MODULE OTRAN       =    34

TOTAL CARDS PRINTED                      =    34

**** LINKING COMPLETED      4/12/84    7:56:26
```

```
           *===========================================================
           *  EQUATES file for the various Hexbus Transfer routines.
           *      HTRAN - cartridge copier
           *      BTRAN - transfer BASIC images using 'SAVE'
           *      OTRAN - transfer BASIC images using 'OLD/RUN'
           *
           *-----------------------------------------------------------
           *  Ram Data buffers for CC-40 transfer routines
           *
     0977  DEV60F EQU  >0977         Device 60 flags
     0976  DEV60C EQU  >0976         Device 60 completed
     0975  DTBUF  EQU  >0975         Command msg data buffer
           *
           *-----------------------------------------------------------
           *  Error status codes for CC-40 transfer routines
           *
     0001  OPTERR EQU  1             Device option error
     0002  ATTERR EQU  2             Attributes error
     0004  DEVNOP EQU  4             Device not open error
     0005  DEVOPN EQU  5             Device already open error
     0008  DATLON EQU  8             Data too long error
     000A  NOTREQ EQU  10            Not requesting service
     000C  BUFERR EQU  12            Buffer size error
     000D  UNSUPP EQU  13            Unsupported command error
     0010  DATERR EQU  16            CRC data failure error
     0020  MFULL  EQU  32            Medium Full
           *
           *-----------------------------------------------------------
           *  I/O bit equates for CC-40 transfer
           *
     0002  LATON  EQU  >02           Latch it
     0082  LATDON EQU  >82           Abort bus operation
     00FD  LATOFF EQU  >FD           Used to toggle lat
     0001  BUSAVL EQU  >01           Bus available bit to test
     000F  MASKD  EQU  >0F           Mask for data nibble
           *
           *-----------------------------------------------------------
           *  HEXBUS Command codes
           *
     0000  OPEN   EQU  0             Open device 60
     0001  CLOSE  EQU  1             Close device 60
     0003  READ   EQU  3             Read from device 60
     0004  WRITE  EQU  4             Write to device 60
     000A  SRVREQ EQU  10            Service request poll
     0040  INPUT  EQU  ?01000000     Input attributes
     0080  OUTPUT EQU  ?10000000     Output attributes
           *
           *-----------------------------------------------------------
           *  I/O equates for CC-40 transfer routines
           *
     0012  DATA   EQU  P18           I/O Bus data
     0013  BAV    EQU  P19           I/O Bus available (bit 0)
     0014  HSK    EQU  P20           I/O Bus handshake (bit 0)
     0014  LAT    EQU  P20           I/O Latch (bit 1)
     0014  LATD   EQU  P20           I/O Latch D (bit 7)
           *
           *-----------------------------------------------------------
```

```
 0058                *  Register equates for CC-40 transfer routines
 0059                *
 0060       003A     TEMPO  EQU   >3A
 0061       0059     MSGPTR EQU   >59               Ptr to command data buffer
 0062       005A     CNT    EQU   >5A               Count of bytes read
 0063       005B     FLG    EQU   >5B               save/ignore byte flag
 0064       005C     ESTAT  EQU   >5C               error status byte
 0065       005E     RLEN   EQU   >5E               response message length
 0066       0060     RDATA  EQU   >60               command data length
 0067       0061     DCODE  EQU   >61               device code storage
 0068       0062     DSTAT  EQU   >62               CRC status
 0069       0063     COMAND EQU   >63               command code
 0070       0067     DATBUF EQU   >67               4 byte return data buffer
 0071       007E     SVRDAT EQU   >7E
 0072       0057     FPSTAK EQU   >57
 0073       0066     STRTAD EQU   >66
 0074       0068     NEWAD  EQU   >68
 0075       006A     MOVLEN EQU   >6A
 0076       0053     CRSPOS EQU   >53
 0077       004C     KBFLGS EQU   >4C
 0078       0055     FRESPC EQU   >55
 0079       004B     FLAGS  EQU   >4B
 0080       0020     RUNMOD EQU   ?00100000
 0081       0010     IMAGE  EQU   ?00010000
 0082                *
 0083                *-----
 0084                * SYSTEM POINTERS USED
 0085                *
 0086       0801     HIRAM  EQU   >0801
 0087       08E9     ASNSTR EQU   >08E9
 0088       08EB     FPBASE EQU   >08EB
 0089       08ED     DYNBAS EQU   >08ED
 0090                *---------------------------------------------------------------
 0091                *  CC-40 System routine equates for transfer routines
 0092                *
 0093       0063     CHKBRK EQU   >0063           equate for CC-40 Check Break rtn
 0094       001E     DSPBUF EQU   >001E           Display 31 chars on display
 0095       004B     OFFCRS EQU   >004B           Turn off cursor
 0096       000C     KEYIN  EQU   >000C           Get a key from keyboard
 0097       0012     TOPLEV EQU   >0012           Go to toplevel
 0098       F836     CALPAG EQU   >F836           equate for CC-40 Call Rom page rtn
 0099       F839     BRPAG  EQU   >F839           equate for CC-40 Branch Rom page
 0100       005D     BEEP   EQU   >005D           BEEP
 0101       F80C     MOVUP  EQU   >F80C           move block of memory up
 0102       0072     TRSHDY EQU   >0072           re-init dynamic memory
 0103       0000     WRTIND EQU   >0000
 0104       0075     CLENUP EQU   >0075
 0105                *
NO ERRORS, NO WARNINGS
```

```
0001                    IDT   'MODHED'
0003                  *
0004                    REF   PHEAD
0005                  *
0006                  *---------------------------------------
0007                  *  Cartridge Header Information
0008                  *
0009 0000    A5         BYTE >A5,>5A              Expected Values
     0001    5A
0010 0002    81         BYTE ?10000001            16K, CC-40, VER 1
0011 0003    9E         BYTE >9E                  immediate execution
0012 0004 0000          DATA PHEAD                ptr to first header
0013 0006 0000          DATA 0                    no basic extensions
0014 0008 0001          DATA PHEAD+1              next available address
0015 000A 0000          DATA 0                    no addr in 5000-8FFF
0016                  *
0017                    END
NO ERRORS, NO WARNINGS
```

```
 0004             *
 0005               IDT   'CONTRL'
 0007             *
 0008               DEF   CTRAN,CONTRL,ABORT,PHEAD
 0009               REF   RCVMSG,RCV
 0010             *
```

```
0012                *
0013                *=====================================================================
0014                *  CTRAN is the main entry point to this transfer package.
0015                *  It initializes the CC40 as a peripheral, and waits for
0016                *  a command message to come across the Hexbus.
0017                *
0018          0000' CTRAN  EQU  $
0019 0000      A2     MOVP %LATON,LAT          init latch
     0001      02
     0002      14
0020 0003      A2     MOVP %BUSAVL,BAV         and set BAV to one
     0004      01
     0005      13
0021 0006      B5     CLR  A                   turn off all the LCD indicators
0022 0007      8B     STA  @>83A
     0008 083A
0023 000A      8B     STA  @>83B
     000B 083B
0024 000D      8B     STA  @>83C
     000E 083C
0025 0010      8B     STA  @>83D
     0011 083D
0026 0013      88     MOVD %WRTIND,B           clear them now
     0014 0000
     0016      01
0027 0017      8E     CALL @CALPAG
     0018 F836
0028 001A      88     MOVD %OFFCRS,B           turn off the cursor
     001B 004B
     001D      01
0029 001E      8E     CALL @CALPAG
     001F F836
0030 0021      D5     CLR  CRSPOS              crspos=0
     0022      53
0031 0023      88     MOVD %CMSG,MSGPTR        display title message
     0024 0139'
     0026      59
0032 0027      88     MOVD %DSPBUF,B
     0028 001E
     002A      01
0033 002B      8E     CALL @CALPAG
     002C F836
0034 002E      88     MOVD %BEEP,B             beep once
     002F 005D
     0031      01
0035 0032      8E     CALL @CALPAG
     0033 F836
0036 0035      73     AND  %>FF-RUNMOD,FLAGS turn off run flag
     0036      DF
     0037      4B
0037 0038      88     MOVD %CLENUP,B           clean up memory
     0039 0075
     003B      01
0038 003C      8E     CALL @CALPAG             if runmod set in clenup, error
     003D F836
0039 003F      74     OR   %RUNMOD,FLAGS       turn run flag back on
     0040      20
     0041   4B
0040                  *  pause for a couple of seconds
0041 0042   22          MOV   %>02,A
     0043   02
0042 0044             LP09
0043 0044   88          PUSH A
0044 0045   88          MOVD %>FFFF,B
     0046 FFFF
     0048   01
0045 0049             LP00
0046 0049   BA          DJNZ A,LP00
     004A   FE
0047 004B   CA          DJNZ B,LP00
     004C   FC
0048 004D   B9          POP   A
0049 004E   BA          DJNZ A,LP09
     004F   F4
0050 0050   88          MOVD %BEEP,B
     0051 005D
     0053   01
0051 0054   8E          CALL @CALPAG
     0055 F836
0052 0057   B5          CLR   A                     reset device 60
0053 0058   8B          STA   @DEV60F               make sure device 60 is closed
     0059 0977
0054 005B   D5          CLR   CRSPOS                cursor in position 0
     005C   53
0055 005D   88          MOVD %SMSG,MSGPTR           display ready message
     005E 011A'
     0060   59
0056 0061   88          MOVD %DSPBUF,B
     0062 001E
     0064   01
0057 0065   8E          CALL @CALPAG
     0066 F836
0058                  *
0059                  * Top of bus message check and wait loop
0060                  *
0061 0068             CTOPL
0062 0068   88          MOVD %CHKBRK,B              look for break key
     0069 0063
     006B   01
0063 006C   8E          CALL @CALPAG
     006D F836
0064 006F   C1          TSTB                        do we see a break?
0065 0070   E2          JZ    NOBRK                 if not, then go check bus out
     0071   30
0066 0072   A2          MOVP %LATDON,LATD           else, clear bus
     0073   82
     0074   14
0067 0075   88          MOVD %BMSG,MSGPTR           set up ptr to termination msg
     0076 00FB'
     0078   59
0068 0079             EXIT01
0069 0079   D5          CLR   CRSPOS                set cursor position to zero
     007A   53
0070 007B   88          MOVD %DSPBUF,B              display termination message
```

```
     007C 001E
     007E   01
0071 007F   8E     CALL @CALPAG
     0080 F836
0072 0082   73     AND  %>F7,KBFLGS        disable FN key processing
     0083   F7
     0084   4C
0073 0085   88     MOVD %>FFFF,B
     0086 FFFF
     0088   01
0074 0089        LP01
0075 0089   CA     DJNZ B,LP01             put a short delay in
     008A   FE
0076 008B   BA     DJNZ A,LP01
     008C   FC
0077 008D   88     MOVD %KEYIN,B           wait for any key
     008E 000C
     0090   01
0078 0091   8E     CALL @CALPAG
     0092 F836
0079 0094   88     MOVD %TRSHDY,B          clean up dynamic memory manage.
     0095 0072
     0097   01
0080 0098   8E     CALL @CALPAG
     0099 F836
0081 009B   88     MOVD %TOPLEV,B          go to top level
     009C 0012
     009E   01
0082 009F   8C     BR   @BRPAG
     00A0 F839
0083             *
0084 00A2        NOBRK
0085 00A2   A6     BTJOP %BUSAVL,BAV,CTOPL if BAV=1 go look for break again
     00A3   01
     00A4   13
     00A5   C2
0086 00A6   8E     CALL @RCV               if BAV=0, msg coming through
     00A7 0000
0087 00A9   B0     TSTA                    see if Device is 0
0088 00AA   E2     JZ   OK                 if so, go for it
     00AB   04
0089 00AC   2D     CMP  %60,A              if device = 60, go for it
     00AD   3C
0090 00AE   E6     JNE  ABORT
     00AF   14
0091 00B0        OK
0092 00B0   D0     MOV  A,DCODE            save device code
     00B1   61
0093 00B2   8A     LDA  @>83A              get I/O indicator status
     00B3 083A
0094 00B5   24     OR   %>04,A             turn on the I/O indicator
     00B6   04
0095 00B7   8B     STA  @>83A
     00B8 083A
0096 00BA   88     MOVD %WRTIND,B          write the indicators
     00BB 0000
     00BD   01
```

```
 0097 00BE   8E    CALL @CALPAG
      00BF F836
 0098 00C1   8C    BR   @RCVMSG             go read the message
      00C2 0000
 0099             *
 0100       00C4' ABORT  EQU  $             Abort bus operation handler
 0101 00C4   A2    MOVP %LATDON,LATD        clear the bus
      00C5   82
      00C6   14
 0102       00C7' CONTRL EQU  $             Re-entry to control loop
 0103 00C7   A7    BTJZP %BUSAVL,BAV,CONTRL Wait for master acknowledge.
      00C8   01
      00C9   13
      00CA   FC
 0104 00CB   8A    LDA  @>83A
      00CC 083A
 0105 00CE   23    AND  %>FB,A              Turn off I/O indicator
      00CF   FB
 0106 00D0   8B    STA  @>83A
      00D1 083A
 0107 00D3   88    MOVD %WRTIND,B           write indicators to display
      00D4 0000
      00D6   01
 0108 00D7   8E    CALL @CALPAG
      00D8 F836
 0109 00DA   8C    BR   @CTOPL              back to top of message read/wait
      00DB 0068'
 0110             *
 0111             *=============================================================
 0112             * Program prompts and messages
 0113             *
 0114             *      Program stopped. Press any key.
 0115 00DD   2E    TEXT '.yek yna sserP .deppots margorP'
      00DE   79
      00DF   65
      00E0   6B
      00E1   20
      00E2   79
      00E3   6E
      00E4   61
      00E5   20
      00E6   73
      00E7   73
      00E8   65
      00E9   72
      00EA   50
      00EB   20
      00EC   2E
      00ED   64
      00EE   65
      00EF   70
      00F0   70
      00F1   6F
      00F2   74
      00F3   73
      00F4   20
      00F5   6D
      00F6   61
      00F7   72
      00F8   67
      00F9   6F
      00FA   72
      00FB   50
0116       00FB' BMSG   EQU  $-1
0117             *      'Running... Press BREAK to stop.'
0118 00FC   2E    TEXT '.pots ot KAERB sserP ...gninnuR'
      00FD   70
      00FE   6F
      00FF   74
      0100   73
      0101   20
      0102   6F
      0103   74
      0104   20
      0105   4B
      0106   41
      0107   45
      0108   52
      0109   42
      010A   20
      010B   73
      010C   73
      010D   65
      010E   72
      010F   50
      0110   20
      0111   2E
      0112   2E
      0113   2E
      0114   67
      0115   6E
      0116   69
      0117   6E
      0118   6E
      0119   75
      011A   52
0119       011A' SMSG   EQU  $-1
0120             *      'CC-40 BASIC Transfer Utility   '
0121 011B   20    TEXT '   ytilitU refsnarT CISAB 04-CC'
      011C   20
      011D   20
      011E   79
      011F   74
      0120   69
      0121   6C
      0122   69
      0123   74
      0124   55
      0125   20
      0126   72
      0127   65
      0128   66
      0129   73
      012A   6E
```

```
          012B   61
          012C   72
          012D   54
          012E   20
          012F   43
          0130   49
          0131   53
          0132   41
          0133   42
          0134   20
          0135   30
          0136   34
          0137   2D
          0138   43
          0139   43
     0122      0139' CMSG   EQU  $-1
     0123            *
     0124            *-------------------------------------------------------------
     0125            *  Program header.  This module must be last in link control
     0126            *
     0127 013A   4E    TEXT 'NARTO'             OTRAN Program
          013B   41
          013C   52
          013D   54
          013E   4F
     0128 013F   05    BYTE >05
     0129      013F' NAME   EQU $-1
     0130 0140 0000'   DATA CTRAN               entry point
     0131 0142 0000    DATA >0000               next link
     0132 0144 FFFC    DATA NAME-$+1            offset to name
     0133 0146   00    BYTE >0                  header length
     0134 0147   C4    BYTE ?11000100           Header flags
     0135      0147' PHEAD  EQU  $-1            address of header
     0136              END
    NO ERRORS, NO WARNINGS
```

```
0004                 *
0005                   IDT   'MESAGE'
0007                 *
0008                   DEF   SNDMSG,RCVMSG
0009                   REF   XMIT,RCV,CONTRL,OPRHND,ABORT
0010                 *
0011                 *=================================================
0012                 *  RCVMSG - reads command message from hexbus and stores it
0013                 *    in the CC-40 Crunch buffer at location >0976 down.
0014                 *    location >0977 is used as the device 60 open/close flag
0015                 *
0016           0000' RCVMSG EQU  $
0017 0000     88      MOVD %DTBUF,MSGPTR      set up pointer to com msg buffer
     0001 0975
     0003     59
0018 0004     D5      CLR  CNT                no bytes read yet
     0005     5A
0019 0006     D5      CLR  ESTAT              clear error status
     0007     5C
0020 0008     88      MOVD %0,RLEN            response length is zero
     0009 0000
     000B     5E
0021 000C     88      MOVD %>FFFF,RDATA       can read up to 64 K bytes
     000D FFFF
     000F     60
0022 0010     8E      CALL @RCV               read command
     0011 0000
0023 0013     D0      MOV  A,COMAND           save it
     0014     63
0024 0015     2D      CMP  %>FF,A             reset command?
     0016     FF
0025 0017     E2      JEQ  RESET              if so, reset
     0018     0C
0026 0019     2D      CMP  %>FE,A             if command is null op, then abort
     001A     FE
0027 001B     E2      JEQ  NULLOP
     001C     0C
0028 001D     7D      CMP  %0,DCODE           if device code is not zero,
     001E     00
     001F     61
0029 0020     E6      JNE  RLOOP              then continue
     0021     0A
0030 0022     8C      BR   @ABORT             else, abort rest of operation
     0023 0000
0031 0025            RESET
0032 0025     B5      CLR  A                  reset consists of storing a zero
0033 0026     8B      STA  @DEV60F            value in the open flags for dev 60
     0027 0977
0034 0029            NULLOP
0035 0029     8C      BR   @ABORT             abort rest of operation
     002A 0000
0036 002C            RLOOP
0037 002C     B8      PUSH A                  Save command code
0038 002D            RLOOP1
0039 002D     8E      CALL @RCV               get byte of command message
     002E 0000
0040 0030     9B      STA  *MSGPTR            save in command buffer
```

```
     0031    59
0041 0032    DB      DECD MSGPTR               adjust to next avail pos in buf
     0033    59
0042 0034    D3      INC  CNT                  add one to count read
     0035    5A
0043 0036    7D      CMP  %7,CNT               have we read in data length yet?
     0037    07
     0038    5A
0044 0039    E6      JNE  NOTNIN               if not, continue
     003A    0A
0045 003B    8A      LDA  @DTBUF-6             else, get data length
     003C  096F
0046 003E    D0      MOV  A,RDATA-1            set up data length remaining
     003F    5F
0047 0040    8A      LDA  @DTBUF-5             get lsb of data length
     0041  0970
0048 0043    D0      MOV  A,RDATA              rdata is remaining data to read
     0044    60
0049 0045          NOTNIN
0050 0045    7D      CMP  %7,CNT
     0046    07
     0047    5A
0051 0048    E7      JL   RLOOP1               if so, continue reading
     0049    E3
0052 004A    72      MOV  %>FE,CNT             if reading data buf, set cnt flag
     004B    FE
     004C    5A
0053 004D    DB      DECD RDATA                use rdata as amount to read
     004E    60
0054 004F    E3      JC   RLOOP1               if not done, continue reading
     0050    DC
0055 0051    8C      BR   @OPRHND              now, go handle the req. operation
     0052  0000
0056               *
0057               *---------------------------------------------------------------
0058               *  SNDMSG - sends out response message on the hexbus.
0059               *    Minimum of three bytes is sent.  Returned data length
0060               *    followed by return status.
0061               *
0062         0054' SNDMSG EQU  $
0063 0054    12      MOV  RLEN,A               get lsb of data length to send
     0055    5E
0064 0056    8E      CALL @XMIT                send it
     0057  0000
0065 0059    12      MOV  RLEN-1,A             get msb of data length to send
     005A    5D
0066 005B    8E      CALL @XMIT                send it
     005C  0000
0067 005E          XLOOP
0068 005E    DB      DECD RLEN                 count to send= cts - 1
     005F    5E
0069 0060    E7      JNC  XDONE                if done, then go send status
     0061    09
0070 0062    9A      LDA  *MSGPTR              else, get data byte
     0063    59
0071 0064    8E      CALL @XMIT                send it
     0065  0000
```

```
 0072 0067    DB     DECD MSGPTR              adjust data ptr
      0068    59
 0073 0069    E0     JMP  XLOOP               continue till done
      006A    F3
 0074 006B          XDONE
 0075 006B    12     MOV  ESTAT,A             get error status
      006C    5C
 0076 006D    8E     CALL @XMIT               send it
      006E  0000
 0077 0070    8C     BR   @CONTRL             go to control loop
      0071  0000
 0078               *
 0079                END
NO ERRORS, NO WARNINGS
```

```
 0004                  *
 0005                    IDT   'OPRHND'
 0007                  *
 0008                    REF   SNDMSG
 0009                    DEF   OPRHND
 0010                  *
 0011                  *=========================================================
 0012                  * OPRHND is the OTRAN Hexbus command interpreter and
 0013                  *  command processor.  It takes the command code given it
 0014                  *  and determines the appropriate action to perform.  This
 0015                  *  action can be anything from opening the device to
 0016                  *  returning an unsupported command error.
 0017                  *
 0018                  *
 0019          0000' OPRHND EQU  $
 0020 0000      7D     CMP   %OPEN,COMAND        Is the command OPEN?
      0001      00
      0002      63
 0021 0003      E2     JEQ   THICMD              if so, then use this processor
      0004      03
 0022 0005      8C     BR    @NXTCMD             else check for the next command
      0006 00A3'
 0023 0008           THICMD
 0024 0008      8A     LDA   @DEV6OF             Get the device open flag
      0009 0977
 0025 000B      E2     JZ    OKOPN               if not open, then proceed
      000C      06
 0026 000D      72     MOV   %DEVOPN,ESTAT       else, return "Device already open"
      000E      05
      000F      5C
 0027 0010      8C     BR    @SNDMSG             send the response
      0011 0000
 0028 0013           OKOPN
 0029 0013      8A     LDA   @DTBUF-5            get the amount of data sent
      0014 0970
 0030 0016      C0     MOV   A,B
 0031 0017      8A     LDA   @DTBUF-6
      0018 096F
 0032 001A      E6     JNZ   BAD                 if more than 256, error
      001B      04
 0033 001C      5D     CMP   %3,B                see if sent more/less than 3
      001D      03
 0034 001E      E2     JEQ   OKDAT               if not, sent proper data
      001F      06
 0035 0020           BAD
 0036 0020      72     MOV   %OPTERR,ESTAT       else return "Options error"
      0021      01
      0022      5C
 0037 0023      8C     BR    @SNDMSG             send the response
      0024 0000
 0038 0026           OKDAT
 0039 0026      8A     LDA   @DTBUF-9            get the open attributes
      0027 096C
 0040 0029      2D     CMP   %INPUT,A            open for input?
      002A      40
 0041 002B      E2     JEQ   OKATT               if so, attributes are ok
      002C      06
```

```
0042 002D   72    MOV  %ATTERR,ESTAT       else return "Attributes error"
     002E   02
     002F   5C
0043 0030   8C    BR   @SNDMSG             send the response
     0031 0000
0044 0033        OKATT
0045 0033   8A    LDA  @HIRAM              get the addr of top of RAM
     0034 0801
0046 0036   D0    MOV  A,DATBUF            this is also ptr to BASIC header
     0037   67
0047 0038   8A    LDA  @HIRAM-1
     0039 0800
0048 003B   D0    MOV  A,DATBUF-1
     003C   66
0049 003D   9A    LDA  *DATBUF             get the program flags
     003E   67
0050 003F   26    BTJO %>40,A,BAD          if program in RAM is assembly, err
     0040   40
     0041   DE
0051 0042   4A    SUB  FRESPC,DATBUF       calc size of program + header len
     0043   55
     0044   67
0052 0045   4B    SBB  FRESPC-1,DATBUF-1
     0046   54
     0047   66
0053 0048   26    BTJO %IMAGE,A,SUPC       is the program compressed image?
     0049   10
     004A   16
0054 004B   98    MOVD FPSTAK,MOVLEN       if not, calc size of Var Name Tab.
     004C   57
     004D   6A
0055 004E   8A    LDA  @ASNSTR             addr of end of VNT
     004F 08E9
0056 0051   C0    MOV  A,B
0057 0052   8A    LDA  @ASNSTR-1
     0053 08E8
0058 0055   4A    SUB  B,MOVLEN            sub from addr of top of VNT
     0056   01
     0057   6A
0059 0058   4B    SBB  A,MOVLEN-1
     0059   00
     005A   69
0060 005B   48    ADD  MOVLEN,DATBUF       add to len of program image
     005C   6A
     005D   67
0061 005E   49    ADC  MOVLEN-1,DATBUF-1
     005F   69
     0060   66
0062 0061        SUPC
0063 0061   8A    LDA  @DTBUF-7            get req. buffer size from buffer
     0062 096E
0064 0064   C0    MOV  A,B
0065 0065   8A    LDA  @DTBUF-8
     0066 096D
0066 0068   C1    TSTB
0067 0069   E6    JNZ  NONZER
     006A   16
```

```
 0068 006B          GOODBF
 0069 006B    88      MOVD %0,DATBUF-2        rec num positioned to is zero
      006C 0000
      006E    65
 0070 006F    88      MOVD %4,RLEN            4 bytes data to return
      0070 0004
      0072    5E
 0071 0073    88      MOVD %DATBUF,MSGPTR     ptr to buffer, acc. buf size is
      0074 0067
      0076    59
 0072 0077    22      MOV  %>FF,A             program len, indicate device is
      0078    FF
 0073 0079    8B      STA  @DEV60F            now open.
      007A 0977
 0074 007C    D5      CLR  ESTAT              no error to return
      007D    5C
 0075 007E    8C      BR   @SNDMSG            send response
      007F 0000
 0076 0081          NONZER
 0077 0081    3A      SUB  DATBUF,B           see if prog will fit in buffer
      0082    67
 0078 0083    1B      SBB  DATBUF-1,A
      0084    66
 0079 0085    E1      JN   NOROOM             if not, return "Buffer size err"
      0086    16
 0080 0087    22      MOV  %>FF,A             set device 60 open
      0088    FF
 0081 0089    8B      STA  @DEV60F
      008A 0977
 0082 008C    88      MOVD %0,DATBUF-2        rec num is zero
      008D 0000
      008F    65
 0083 0090    88      MOVD %DATBUF,MSGPTR     ptr to data to return
      0091 0067
      0093    59
 0084 0094    88      MOVD %4,RLEN            return 4 bytes
      0095 0004
      0097    5E
 0085 0098    D5      CLR  ESTAT              normal operation completion
      0099    5C
 0086 009A    8C      BR   @SNDMSG            send the response
      009B 0000
 0087 009D          NOROOM
 0088 009D    72      MOV  %BUFERR,ESTAT      buffer size error
      009E    0C
      009F    5C
 0089 00A0    8C      BR   @SNDMSG            send it
      00A1 0000
 0090               *
 0091 00A3          NXTCMD
 0092 00A3    7D      CMP  %CLOSE,COMAND      is the command CLOSE?
      00A4    01
      00A5    63
 0093 00A6    E6      JNE  RDCMD              if not, check for read command
      00A7    1F
 0094 00A8    8A      LDA  @DEV60F            see if device 60 is open
      00A9 0977
```

```
0095 00AB   E6      JNZ  OPEN60              if so, continue
     00AC   06
0096 00AD   72      MOV  %DEVNOP,ESTAT       else, give the device not open
     00AE   04
     00AF   5C
0097 00B0   8C      BR   @SNDMSG             error return
     00B1 0000
0098 00B3         OPEN60
0099 00B3   B5      CLR  A                   make sure device is closed
0100 00B4   8B      STA  @DEV60F
     00B5 0977
0101 00B7   88      MOVD %TRSHDY,B           init dynamic memory
     00B8 0072
     00BA   01
0102 00BB   8E      CALL @CALPAG
     00BC F836
0103 00BE   88      MOVD %0,RLEN             zero data in response message
     00BF 0000
     00C1   5E
0104 00C2   D5      CLR  ESTAT               normal completion
     00C3   5C
0105 00C4   8C      BR   @SNDMSG             send the response
     00C5 0000
0106              *
0107      00C7' RDCMD EQU  $
0108 00C7   7D      CMP  %READ,COMAND        see if command is read data
     00C8   03
     00C9   63
0109 00CA   E6      JNE  FINCOM              if not, check for last command
     00CB   48
0110 00CC         READ1
0111 00CC   8A      LDA  @DEV60F             see if device is open
     00CD 0977
0112 00CF   E6      JNZ  OKR1                if so, continue with operation
     00D0   06
0113 00D1   72      MOV  %DEVNOP,ESTAT       else, return "Device not open"
     00D2   04
     00D3   5C
0114 00D4   8C      BR   @SNDMSG             send the response
     00D5 0000
0115 00D7         OKR1
0116 00D7   8A      LDA  @HIRAM              get ptr to top of program
     00D8 0801
0117 00DA   C0      MOV  A,B
0118 00DB   8A      LDA  @HIRAM-1
     00DC 0800
0119 00DE   98      MOVD B,MSGPTR            set response pointer to it
     00DF   01
     00E0   59
0120 00E1   9A      LDA  *B                  get program flags
     00E2   01
0121 00E3   26      BTJO %IMAGE,A,SUPCRN     if compressed, skip VNT move
     00E4   10
     00E5   1C
0122 00E6   8A      LDA  @ASNSTR             else move up the VNT
     00E7 08E9
0123 00E9   C0      MOV  A,B
```

```
 0124 00EA    8A     LDA   @ASNSTR-1
      00EB  08E8
 0125 00ED    98     MOVD  FPSTAK,MOVLEN       calc size of VNT to move
      00EE    57
      00EF    6A
 0126 00F0    4A     SUB   B,MOVLEN
      00F1    01
      00F2    6A
 0127 00F3    4B     SBB   A,MOVLEN-1
      00F4    00
      00F5    69
 0128 00F6    98     MOVD  MOVLEN,RLEN         put VNT size in response len
      00F7    6A
      00F8    5E
 0129 00F9    98     MOVD  FRESPC,NEWAD        set up memory move
      00FA    55
      00FB    68
 0130 00FC    98     MOVD  FPSTAK,STRTAD
      00FD    57
      00FE    66
 0131 00FF    8E     CALL  @MOVUP              move up the VNT
      0100  F80C
 0132 0102         SUPCRN
 0133 0102    98     MOVD  MSGPTR,B            get ptr to top of program
      0103    59
      0104    01
 0134 0105    3A     SUB   FRESPC,B            calc program size
      0106    55
 0135 0107    1B     SBB   FRESPC-1,A
      0108    54
 0136 0109    48     ADD   B,RLEN              add size to response length
      010A    01
      010B    5E
 0137 010C    49     ADC   A,RLEN-1
      010D    00
      010E    5D
 0138 010F    D5     CLR   ESTAT               normal completion of operation
      0110    5C
 0139 0111    8C     BR    @SNDMSG             send the data response message
      0112  0000
 0140              *
 0141 0114         FINCOM
 0142 0114    7D     CMP   %SRVREQ,COMAND      see if service request poll
      0115    0A
      0116    63
 0143 0117    E6     JNE   UNSUPC              if not, return unsupported command
      0118    06
 0144 0119    72     MOV   %NOTREQ,ESTAT       else return not requesting service
      011A    0A
      011B    5C
 0145 011C    8C     BR    @SNDMSG             send the response
      011D  0000
 0146 011F         UNSUPC
 0147 011F    72     MOV   %UNSUPP,ESTAT       unsupported command error
      0120    0D
      0121    5C
 0148 0122    8C     BR    @SNDMSG             send the response
```

```
      0123 0000
 0149              *
 0150                 END
NO ERRORS, NO WARNINGS
```

```
 0004                 *
 0005                   IDT   'XMTRCV'
 0007                 *
 0008                   DEF   XMIT,RCV
 0009                   REF   ABORT
 0010                 *
 0011                 *-----------------------------------------------------
 0012                 * XMIT transmits 8-bit value in A register on the
 0013                 *   Hex-bus.  Least significant nibble is transmitted
 0014                 *   first, followed by the Most significant nibble.
 0015                 *
 0016           0000' XMIT   EQU  $
 0017 0000      A6      BTJOP %BUSAVL,BAV,ABRT1 if BAV=1 abort the operation
      0001      01
      0002      13
      0003      26
 0018 0004      A4      ORP   %LATON,LAT          set LAT to high
      0005      02
      0006      14
 0019 0007      A7      BTJZP %1,HSK,XMIT         wait for handshake
      0008      01
      0009      14
      000A      F5
 0020 000B      82      MOVP  A,DATA              ok, send LSNibble, its ready
      000C      12
 0021 000D      A3      ANDP  %LATOFF,LAT         LAT = 0
      000E      FD
      000F      14
 0022 0010      B0      TSTA                      bus timing (~8 us)
 0023 0011      A4      ORP   %LATON,LAT          LAT = 1 (toggle LAT)
      0012      02
      0013      14
 0024 0014      B7      SWAP  A                   get MSNibble of data
 0025 0015            XMIT2
 0026 0015      A6      BTJOP %BUSAVL,BAV,ABRT1 if BAV=1 abort the operation
      0016      01
      0017      13
      0018      11
 0027 0019      A4      ORP   %LATON,LAT          set LAT to high
      001A      02
      001B      14
 0028 001C      A7      BTJZP %1,HSK,XMIT2        wait for handshake
      001D      01
      001E      14
      001F      F5
 0029 0020      82      MOVP  A,DATA              send LSNibble of data when ready
      0021      12
 0030 0022      A3      ANDP  %LATOFF,LAT         toggle LAT again
      0023      FD
      0024      14
 0031 0025      B0      TSTA                      timing wait (~8 us)
 0032 0026      A4      ORP   %LATON,LAT          set LAT to 1
      0027      02
      0028      14
 0033 0029      0A      RETS                      return
 0034                 *
 0035                 *-----------------------------------------------------
 0036                * ABRT1 - aborts the current I/O operation.
 0037                *
 0038 002A           ABRT1
 0039 002A   B9        POP    A                    throw away return address
 0040 002B   B9        POP    A                    better only be one addr on stk
 0041 002C   8C        BR     @ABORT               go to abort handler
      002D 0000
 0042                *
 0043                *--------------------------------------------------------
 0044                * RCV - Receive 8 bit value from Hex-bus, returns it in
 0045                *    A register.
 0046                *
 0047      002F' RCV    EQU  $
 0048 002F   A6        BTJOP %BUSAVL,BAV,ABRT1 if BAV=1 then abort
      0030   01
      0031   13
      0032   F7
 0049 0033   A6        BTJOP %LATON,LAT,RCV     if LAT=1 then wait
      0034   02
      0035   14
      0036   F8
 0050 0037   91        MOVP   DATA,B               read LSN of data
      0038   12
 0051 0039   53        AND    %MASKD,B             turn off MSN of garbage
      003A   0F
 0052 003B   A2        MOVP   %LATON,LAT           LAT=1
      003C   02
      003D   14
 0053 003E           RCV2
 0054 003E   A6        BTJOP %BUSAVL,BAV,ABRT1 if BAV=1 then abort
      003F   01
      0040   13
      0041   E8
 0055 0042   A6        BTJOP %LATON,LAT,RCV2    if LAT=1 then wait
      0043   02
      0044   14
      0045   F8
 0056 0046   80        MOVP   DATA,A               read MSN of data
      0047   12
 0057 0048   23        AND    %MASKD,A             turn off garbage
      0049   0F
 0058 004A   A2        MOVP   %LATON,LAT           LAT=1
      004B   02
      004C   14
 0059 004D   B7        SWAP   A                    move nibble to high 4 bits
 0060 004E   64        OR     B,A                  put LSN of data in
 0061 004F   0A        RETS                        return with data byte
 0062                *
 0063                  END
NO ERRORS, NO WARNINGS
```